

CLAIMS:

1 1. A method for providing dynamic feedback about a workload of a service provider
2 in a switched fabric, comprising:

3 receiving an incoming service request from a client, via said switched fabric;

4 determining a current workload of the service provider; and

5 sending a response message based on the current workload of the service provider to the
6 client, via said switched fabric, including information needed by the client to issue future service
7 requests to prevent unnecessary timeouts and retries of duplicate service requests.

8 2. The method as claimed in claim 1, wherein the information included in the
9 response message contains a response time value (*RespTimeValue*) indicating the amount of time
10 the service provider expects to respond to the incoming service request based on the workload of
11 the service provider.

12 3. The method as claimed in claim 2, wherein the current workload indicates a
13 number of service requests pending in a service queue of the service provider to be processed by
14 the service provider.

1 4. The method as claimed in claim 1, wherein the client and the service provider
2 correspond to a respective end node of said switched fabric having one or more channel adapters
3 installed to support one or more ports for data communication via physical links of said switched
4 fabric.

5 5. The method as claimed in claim 2, wherein, when the current workload exceeds a
6 certain level, the service provider sets a flag indicating that no new service requests will be
7 accepted for processing, and generates the response message including a larger response time
8 value (*RespTimeValue*) to inform the client that the service provider is overloaded and needs
9 more time to process service requests.

1 6. The method as claimed in claim 2, wherein, when the current workload falls
2 below a certain level, the service provider sets a flag indicating that it is lightly loaded, and
3 generates the response message including a smaller response time value (*RespTimeValue*) to
4 inform the client that the service provider is not overloaded and needs less time to process
5 service requests.

1 7. The method as claimed in claim 1, wherein, when determining the current

workload of the service provider, the service provider performs:

checking if a *DeferNewRequest* flag is set to TRUE indicating that the service provider is overloaded;

if the *DeferNewRequest* flag is set to TRUE, scheduling and sending the response message to the client, including a response time value (*RespTimeValue*) that corresponds to an overload state of the service provider;

if the *DeferNewRequest* flag is set to FALSE, accepting the incoming service request from the client in a service queue for processing;

determining if the incoming service request posted at the service queue makes the service queue to go over a high watermark;

if the incoming service request posted at the service queue makes the service queue to go over the high watermark, computing a larger value for *RespTimeValue*, setting the *DeferNewRequest* flag to TRUE, and proceeding to process the incoming service request posted at the service queue; and

if the incoming service request posted at the service queue does not make the service queue to go over the high watermark, proceeding to process the incoming service request posted at the service queue.

8. The method as claimed in claim 7, wherein the service provider picks up service

1 requests to process by:

2 removing a service request at the head of the service queue for processing and
3 decrementing a count of pending service requests at the service queue;

4 determining if a count of pending service requests equals to a high watermark indicating
5 an overload state;

6 if a count of pending service requests at the service queue equals to the high watermark,
setting the *DeferNewRequest* flag to FALSE to allow all future service requests to be processed,
and proceeding to process the incoming service request posted at the service queue;

7 if a count of pending service requests at the pending service queue is not equal to the high
8 watermark, determining if a count of pending service equals to a low watermark indicating a
9 virtually idle state;

10 if the count of pending service requests equals to the low watermark, computing a smaller
11 value for *RespTimeValue* and proceeding to process the incoming service request posted at the
12 service queue; and

13 if the count of pending service requests is not equal to the low watermark, proceeding to
14 process the incoming service request posted at the service queue.

1 9. The method as claimed in claim 8, wherein the client is further configured to
2 increase or decrease timeout values to prevent unnecessary timeouts and retries of duplicate

1 service requests.

1 10. The method as claimed in claim 1, wherein the client and the service provider are
2 configured in accordance with the *"InfiniBand™ Architecture Specification"*.

1 11. A data network, comprising:

2 a service provider having at least one channel adapter (CA) installed therein supporting
one or more ports;

3 at least one client having at least one channel adapter (CA) installed therein supporting
one or more ports; and

4 a switched fabric comprising a plurality of different switches which interconnect the
5 service provider via CA ports to the client via CA port along different physical links for data
6 communications,
7

8 wherein the service provider further includes a dynamic workload feedback mechanism
9 configured to provide dynamic feedback to the client about a workload of the service provider for
10 preventing unnecessary timeouts and retries of duplicate service requests.
11

1 12. The data network as claimed in claim 11, wherein the dynamic workload feedback

1 mechanism is configured to determine a current workload of the service provider upon receipt of
2 an incoming service request from a client, via said switched fabric; and send a response message
3 based on the current workload of the service provider to the client, via said switched fabric,
4 including information needed by the client to issue future service requests to prevent unnecessary
5 timeouts and retries of duplicate service requests.

13. The data network as claimed in claim 12, wherein the information included in the
response message contains a response time value (*RespTimeValue*) indicating the amount of time
the service provider expects to respond to the incoming service request based on the workload of
the service provider.

14. The data network as claimed in claim 12, wherein the current workload indicates a
number of service requests pending in a service queue of the service provider to be processed by
the service provider.

15. The data network as claimed in claim 13, wherein, when the current workload
exceeds a certain level, the service provider sets a flag indicating that no new service requests
will be accepted for processing, and generates the response message including a larger response
time value (*RespTimeValue*) to inform the client that the service provider is overloaded and

1 needs more time to process service requests.

1 16. The data network as claimed in claim 13, wherein, when the current workload
2 falls below a certain level, the service provider sets a flag indicating it is lightly loaded, and
3 generates the response message including a smaller response time value (*RespTimeValue*) to
4 inform the client that the service provider is not overloaded and needs less time to process
service requests.

17. The data network as claimed in claim 12, wherein the dynamic workload feedback
mechanism is activated to perform the following:

checking if a *DeferNewRequest* flag is set to TRUE indicating that the service provider is
overloaded;

if the *DeferNewRequest* flag is set to TRUE, scheduling and sending the response
message to the client, including a response time value (*RespTimeValue*) that corresponds to an
overload state of the service provider;

if the *DeferNewRequest* flag is set to FALSE, accepting the incoming service request
from the client in a service queue for processing;

determining if the incoming service request posted at the service queue makes the service
queue to go over a high watermark;

1 if the incoming service request posted at the service queue makes the service queue to go
2 over the high watermark, computing a larger value for *RespTimeValue*, setting the
3 *DeferNewRequest* flag to TRUE, and proceeding to process the incoming service request posted
4 at the service queue; and

5 if the incoming service request posted at the service queue does not make the service
6 queue to go over the high watermark, proceeding to process the incoming service request posted
at the service queue.

18. The data network as claimed in claim 17, wherein the dynamic workload feedback
mechanism is activated to further perform the following:

removing a service request at the head of the service queue for processing and
decrementing a count of pending service requests at the service queue;

5 determining if a count of pending service requests equals to a high watermark indicating
6 an overload state;

7 if a count of pending service requests at the service queue equals to the high watermark,
8 setting the *DeferNewRequest* flag to FALSE to allow all future service requests to be processed,
9 and proceeding to process the incoming service request posted at the service queue;

10 if a count of pending service requests at the pending service queue is not equal to the high
11 watermark, determining if a count of pending service equals to a low watermark indicating a

1 virtually idle state;

2 if the count of pending service requests equals to the low watermark, computing a smaller
3 value for *RespTimeValue* and proceeding to process the incoming service request posted at the
4 service queue; and

5 if the count of pending service requests is not equal to the low watermark, proceeding to
6 process the incoming service request posted at the service queue.

19. The data network as claimed in claim 18, wherein the client is further configured
to increase or decrease timeout values to prevent unnecessary timeouts and retries of duplicate
service requests.

20. A computer readable medium comprising instructions that, when executed by a
host system in a switched fabric including end nodes and switches interconnected via links, cause
the host system to:

determine a current workload of the host system, upon receipt of an incoming service
request from a client node, via said switched fabric; and

send a response message based on the current workload of the host system to the client
node, via said switched fabric, including information needed by the client node to issue future
service requests to prevent unnecessary timeouts and retries of duplicate service requests.

1 21. The computer readable medium as claimed in claim 19, wherein the information
2 included in the response message contains a response time value (*RespTimeValue*) indicating the
3 amount of time the host system expects to respond to the incoming service request based on the
4 workload of the host system.

22. The computer readable medium as claimed in claim 19, wherein the current
workload indicates a number of service requests pending in a service queue of the host system to
be processed by the host system.

23. The computer readable medium as claimed in claim 19, wherein, when the current
workload exceeds a certain level, the host system sets a flag indicating that no new service
3 requests will be accepted for processing, and generates the response message including a larger
4 response time value (*RespTimeValue*) to inform the client node that the host system is overloaded
5 and needs more time to process service requests.

1 24. The computer readable medium as claimed in claim 19, wherein, when the current
2 workload falls below a certain level, the host system sets a flag indicating that new service
3 requests will be accepted for processing, and generates the response message including a smaller

- 1 response time value (*RespTimeValue*) to inform the client node that the host system is not
- 2 overloaded and needs less time to process service requests.